

# The RStudio IDE

*using RStudio to manage code, data, and files*

by Martin Frigaard

Written: September 30 2021

Updated: November 24 2021

[Created using the "λέξις" theme](#)

# The RStudio IDE

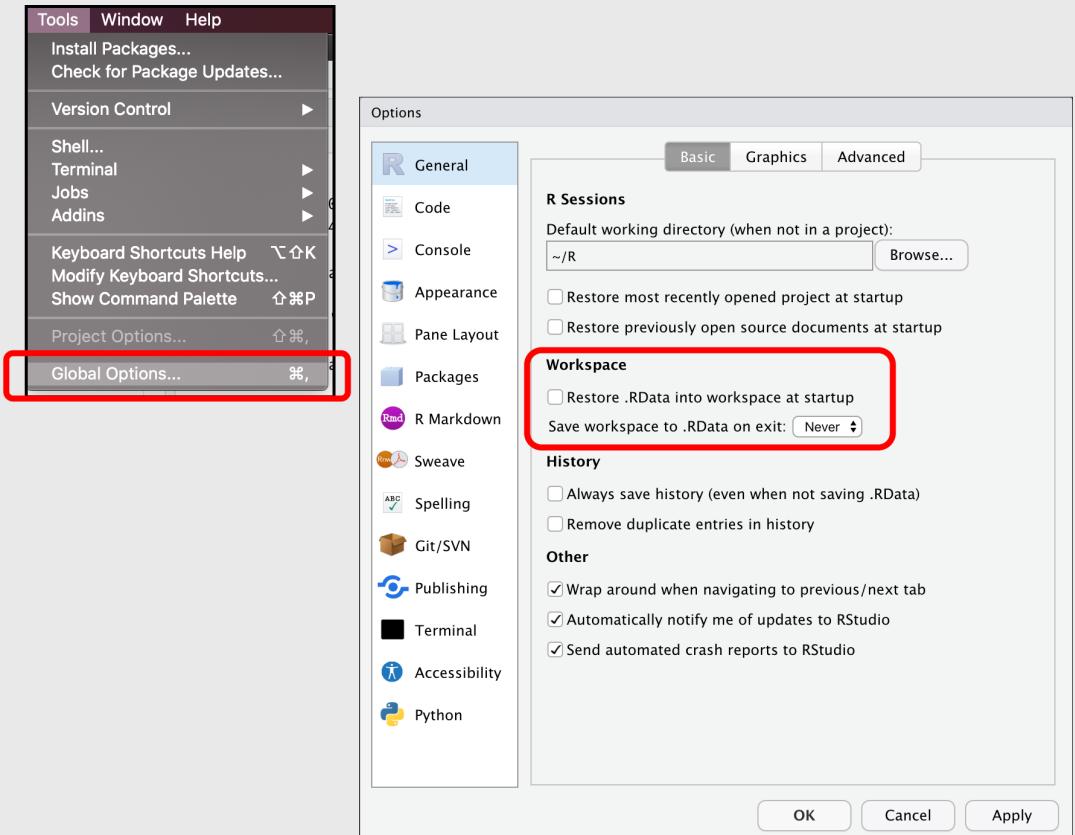
Managing your code, data, and files with RStudio

# Change RStudio's Default Settings

**Click on Tools > Global Options...**

- We want to uncheck "*Restore .RData into workspace at start up*"

- We also want to make sure we change "*Save workspace to .Rdata on exit*" to "*Never*"



# Customize RStudio

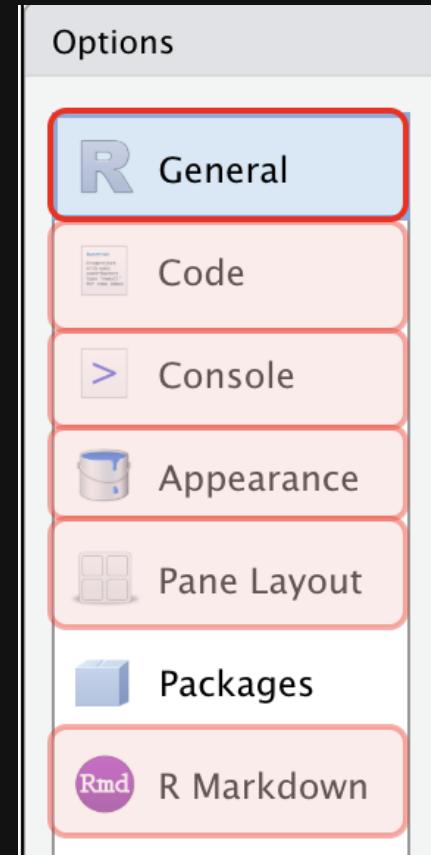
Code

Console

Appearance

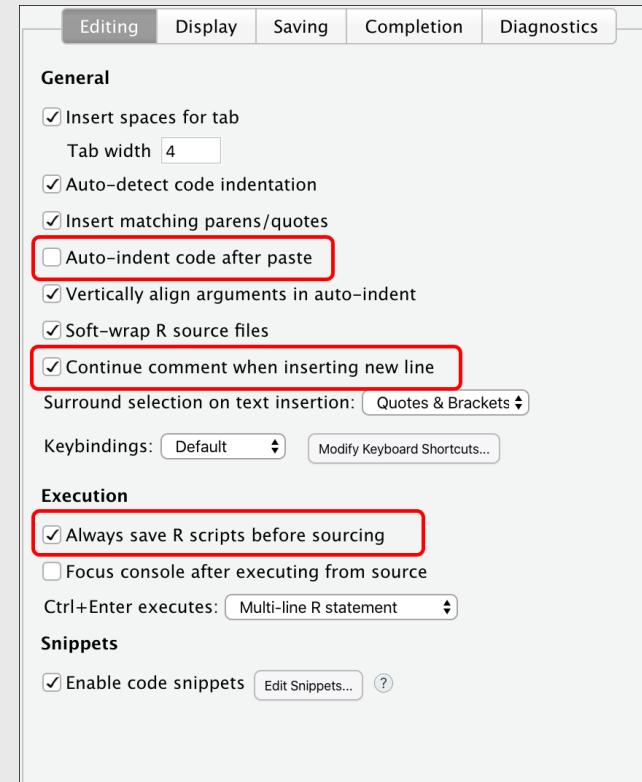
Pane Layout

R Markdown



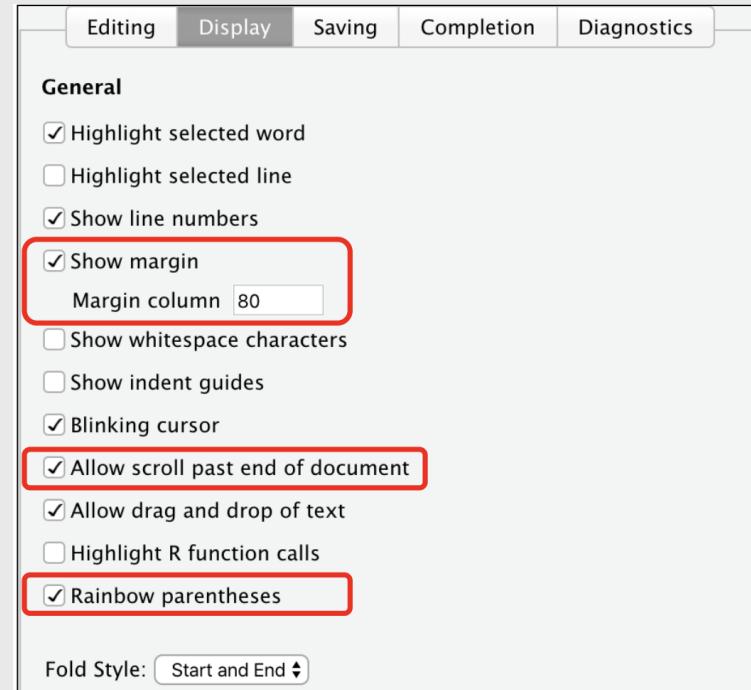
# Code Editing

- Auto indent?
- Continue comment lines?
- Save R scripts before sourcing?



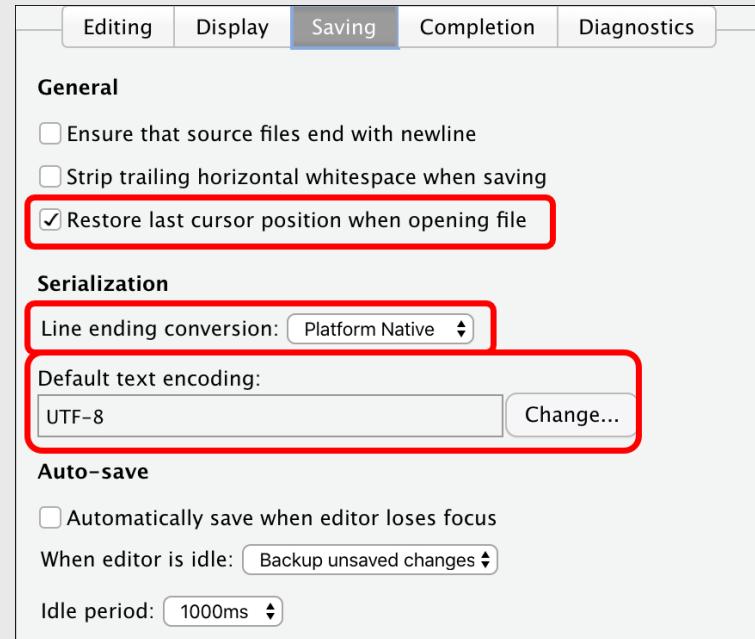
# Code Display

- Margins?
- Scrolling?
- Rainbow parentheses?



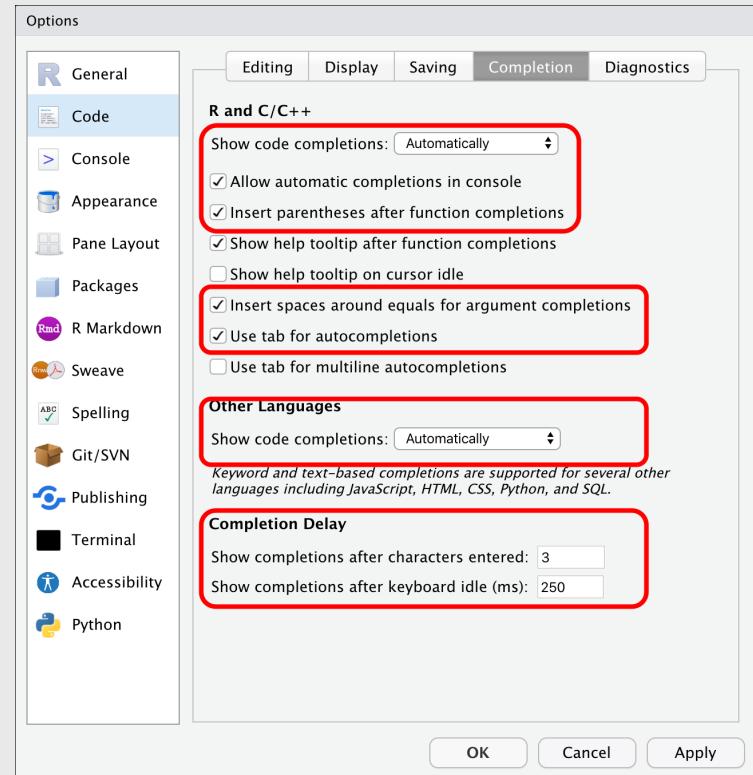
# Code Saving

- Cursor position?
- Line endings?
- Text encoding?



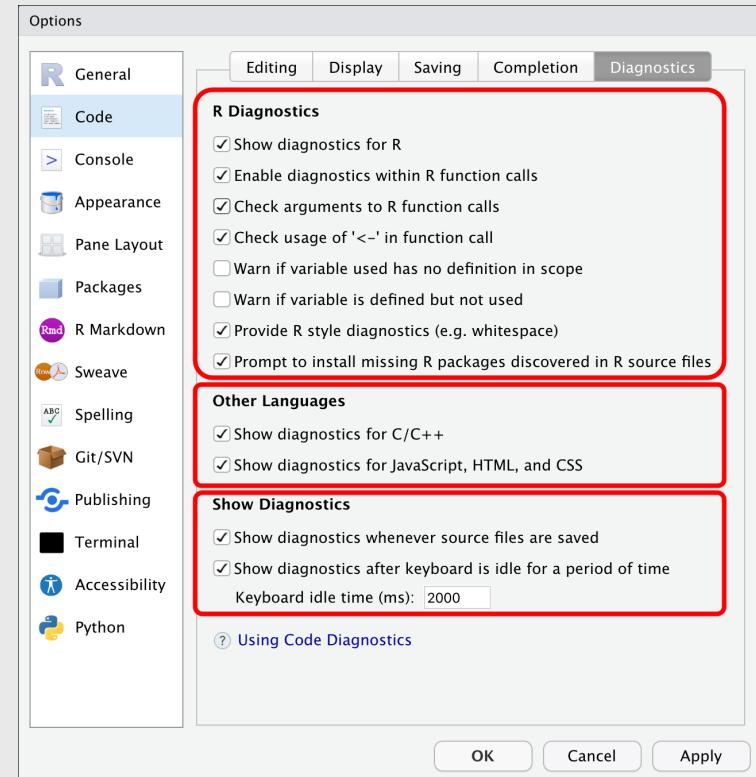
# Code Completion

- Insert parentheses?
- Insert spaces?
- Completion delay setting?



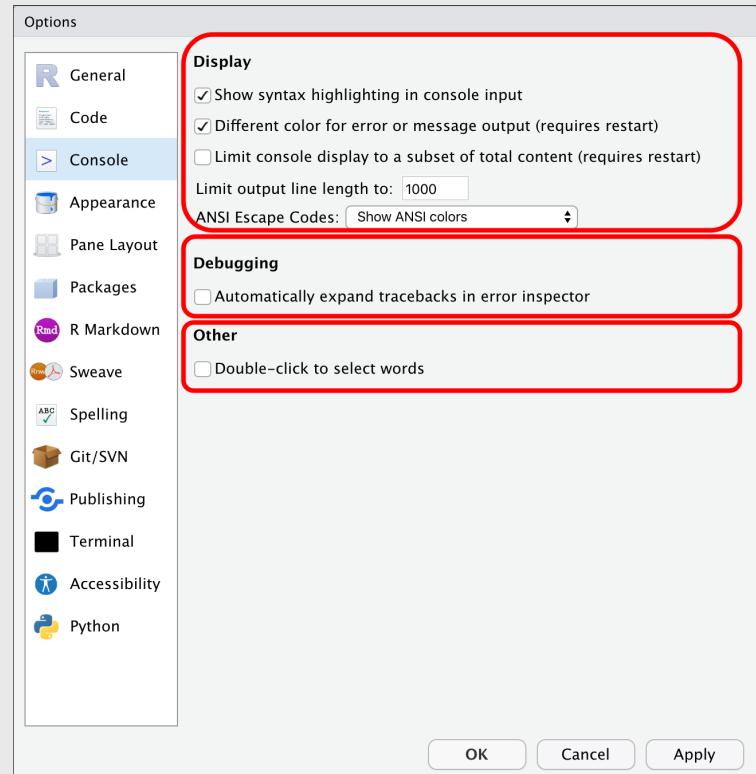
# Code Diagnostics

- Check your R Code?
- Check other languages?
- How long?



# Console

- Display?
- Debugging?
- Other?



# Appearance

- RStudio theme?

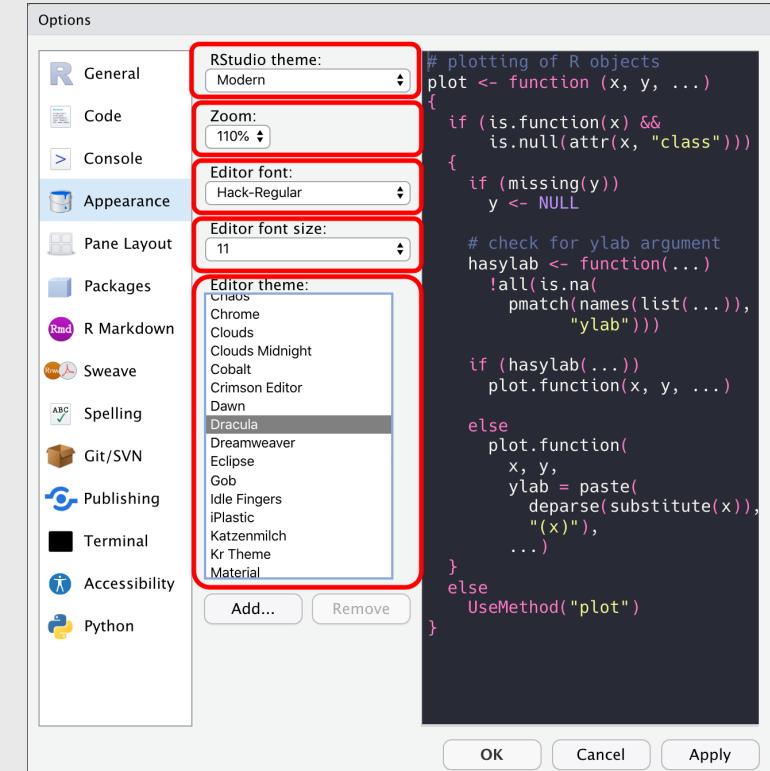
- Zoom?

- Also hold ⌘ and press + on macOS

- Also hold ctrl and press + on Windows

- Font?

- Editor theme?



# Pane layout

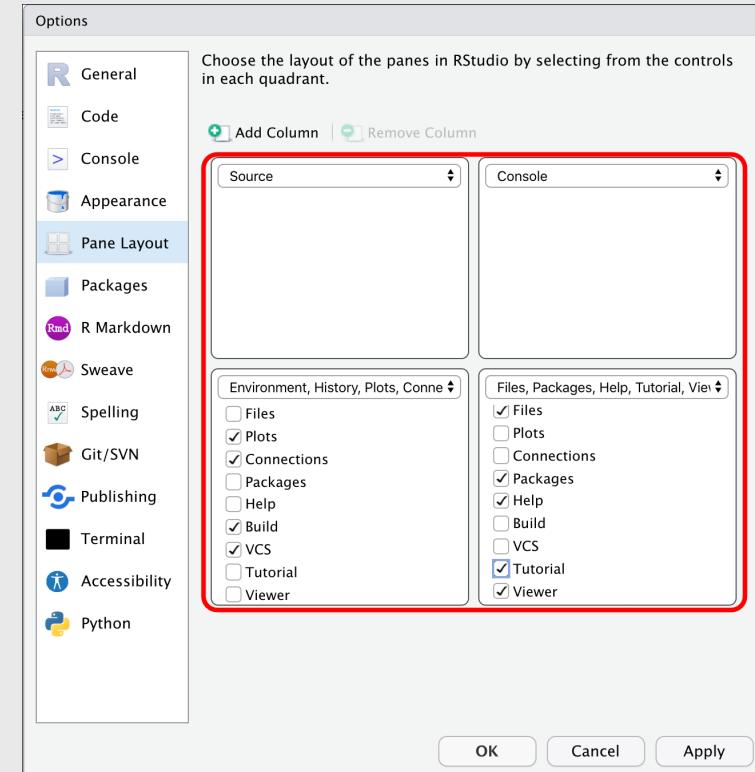
- Source?

- Console?

## Combining pane elements?

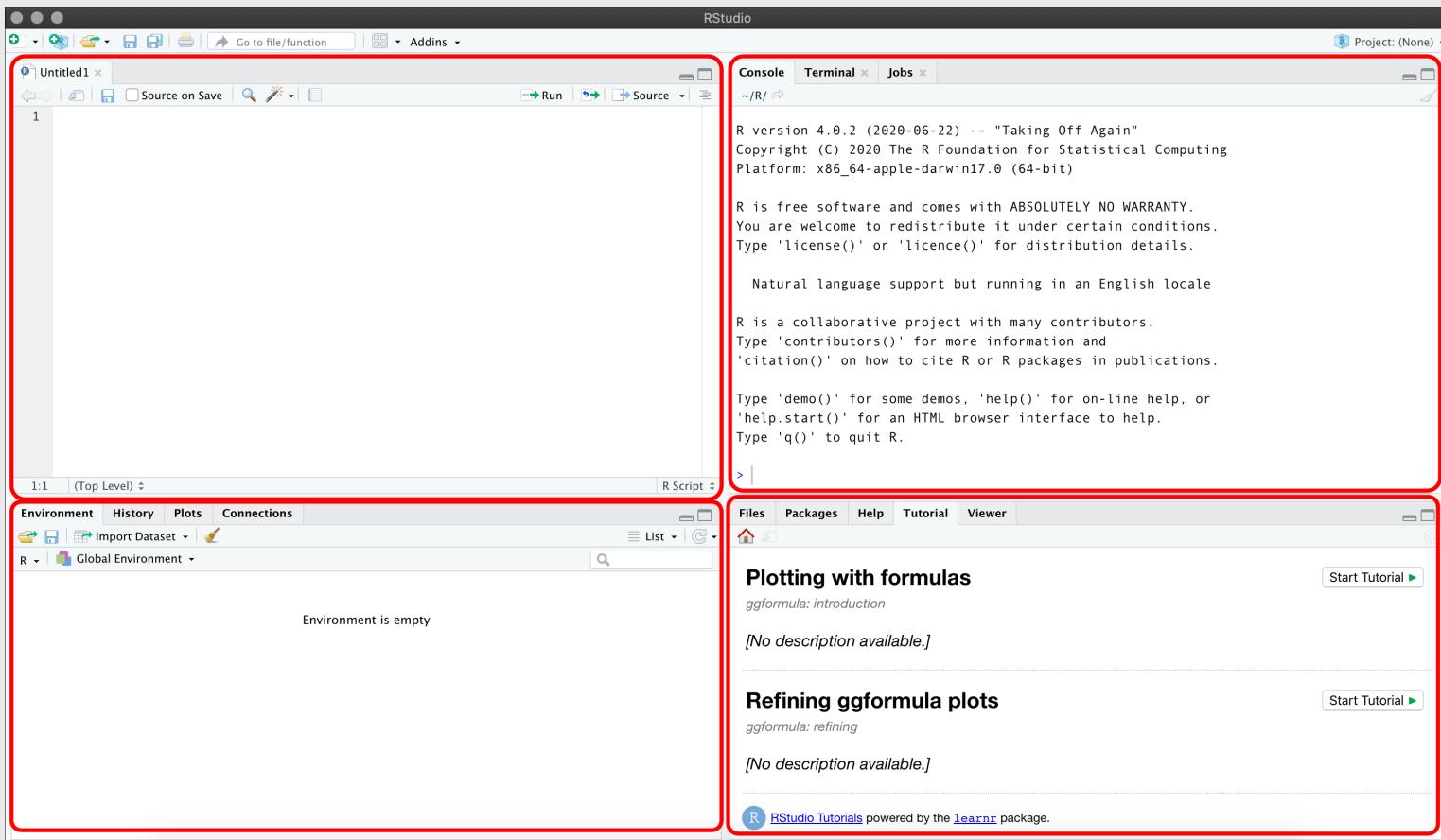
- Plots, Connections, Build, VCS, Presentation

- Files, Packages, Help, Tutorial, Viewer



# Pane layout view

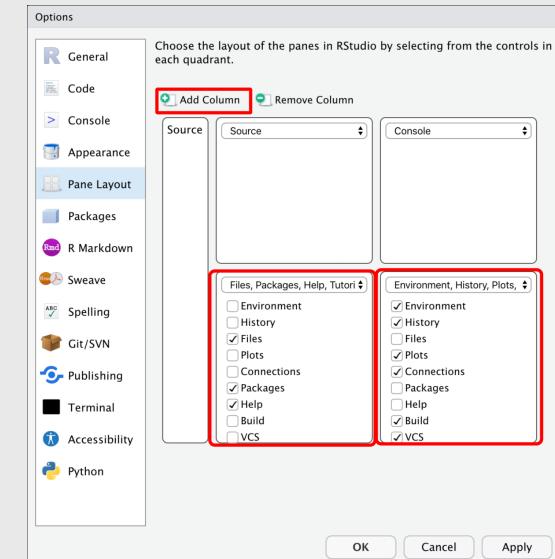
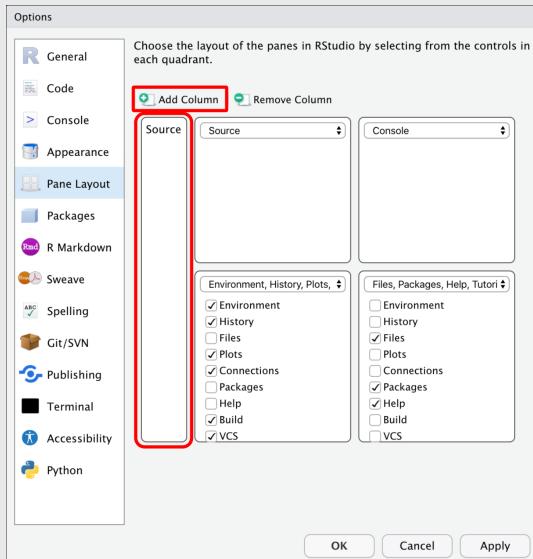
## Standard layout options



# Pane layout: add column

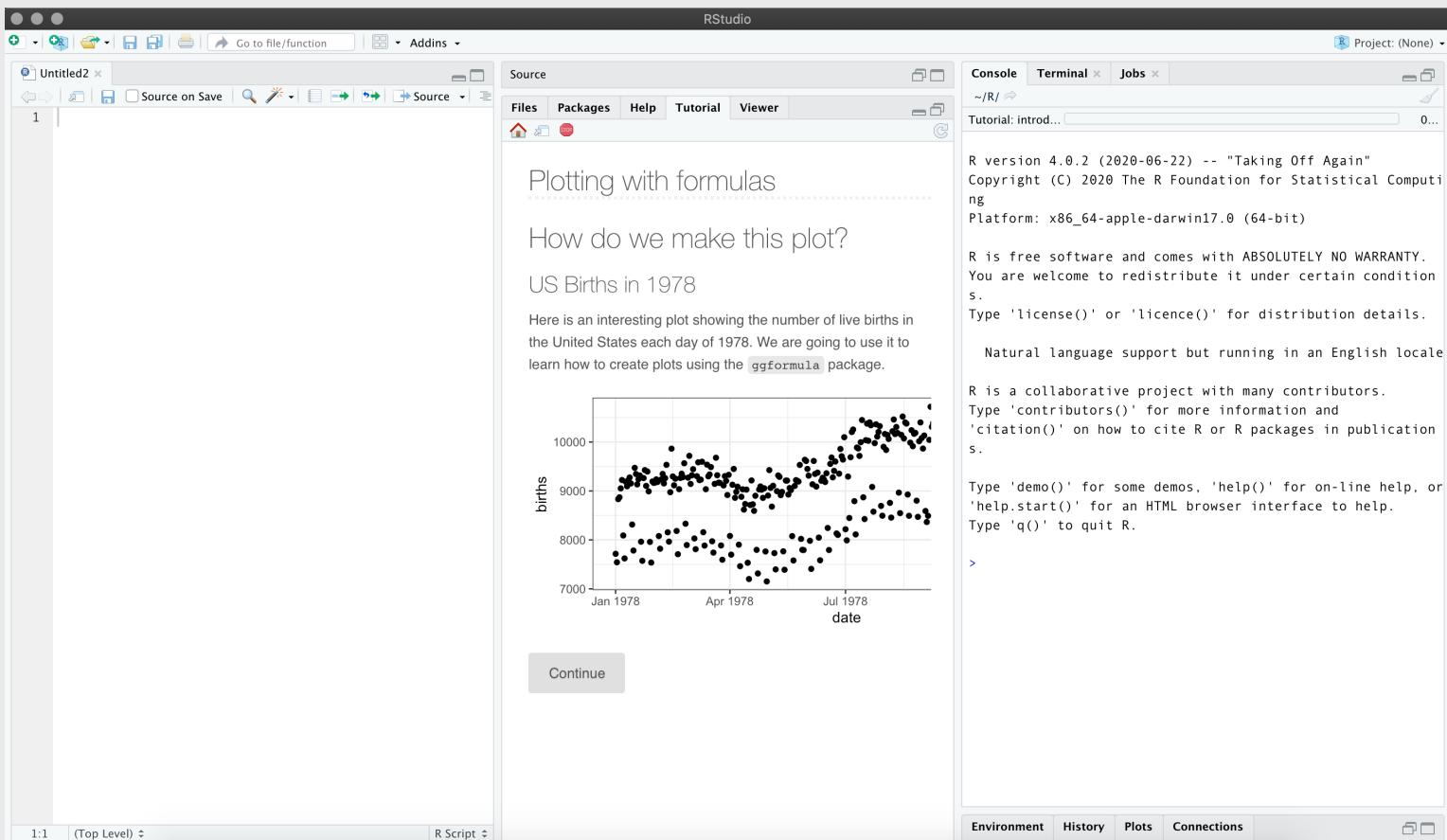
Two screens?

- add a Source column and rearrange the panes



# Pane layout: add column view

Now you see **Source**, **Tutorial**, and **Console** panes on a single screen!



# RStudio Projects

# Why RStudio Projects?

Keep track of all your files with RStudio project files (`.Rproj`).

## Self contained

Using R projects keeps track or your current working directory!

## Project orientated

`.Rproj` files make bundling and shipping files and folders easier!

# Why RStudio Projects?

Keep track of all your files with RStudio project files (`.Rproj`).

**Avoid removing all the files**

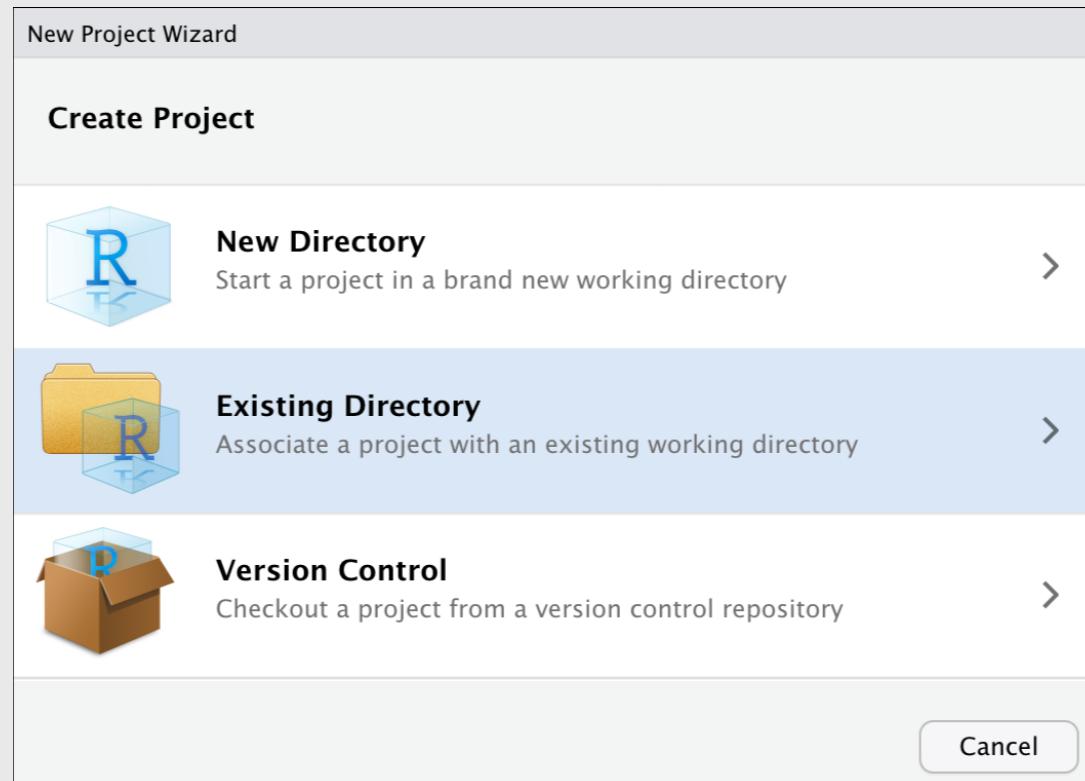
**Don't do this!**

```
rm(list = ls())
```

`.Rproj` files keep all the files associated with a project together – input data, R scripts, analytic results, figures.

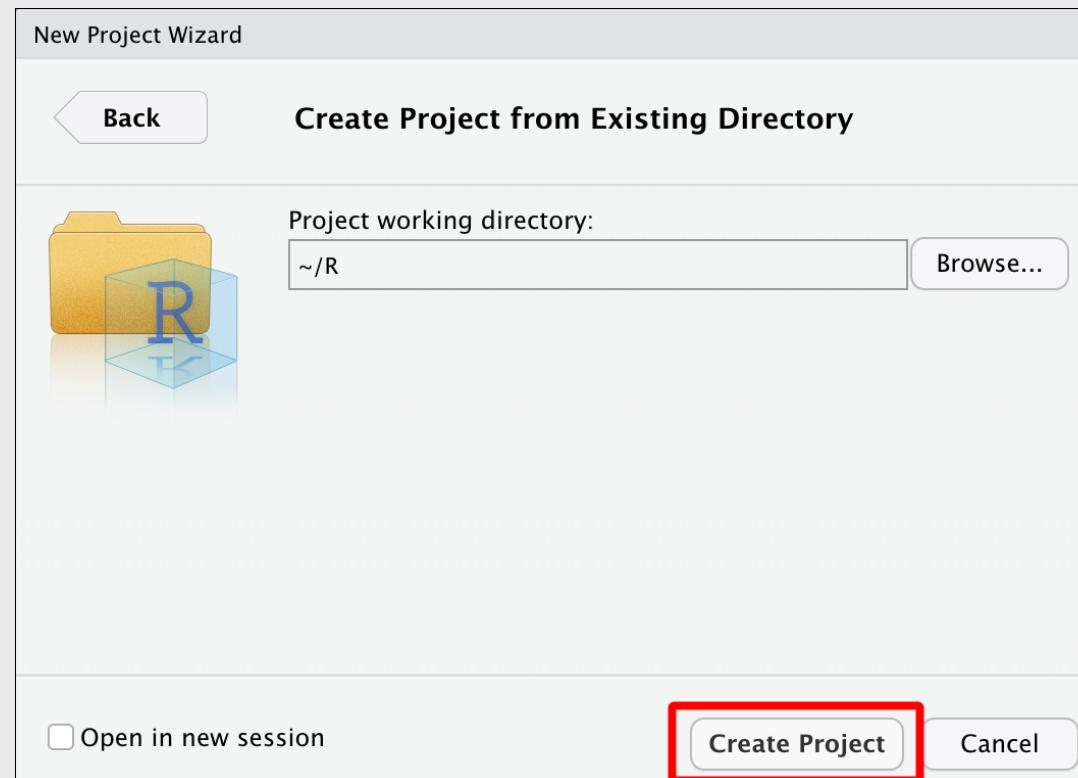
# Creating RStudio project in existing folder

**Click on 'Project: (None)' > 'New Project'**



# Creating RStudio project in existing folder

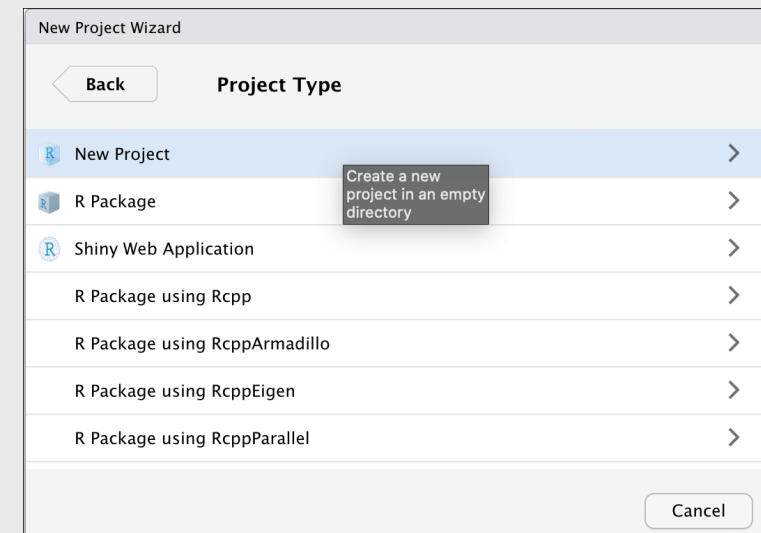
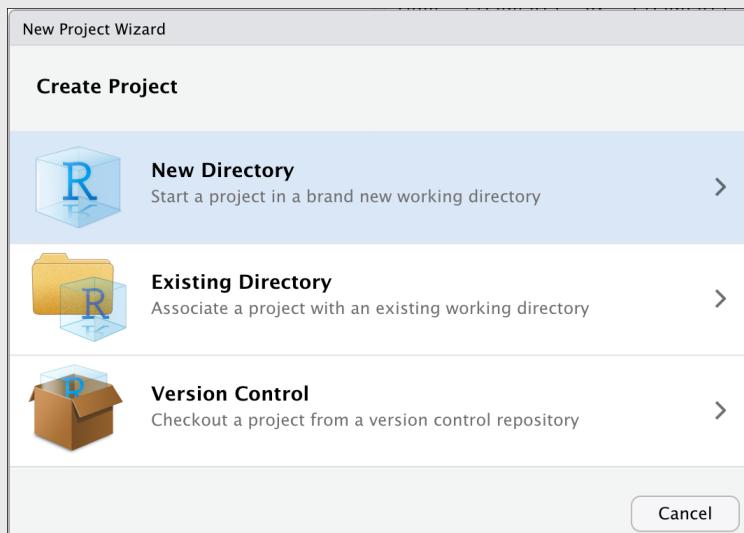
**Click on 'Browse > 'Create Project'**



# Creating RStudio projects in new folder

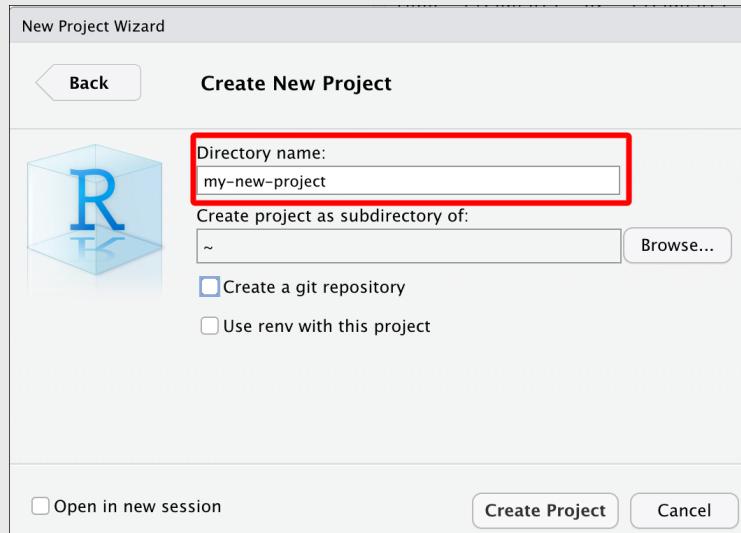
Click on Project: (None) > New Project

Select project type

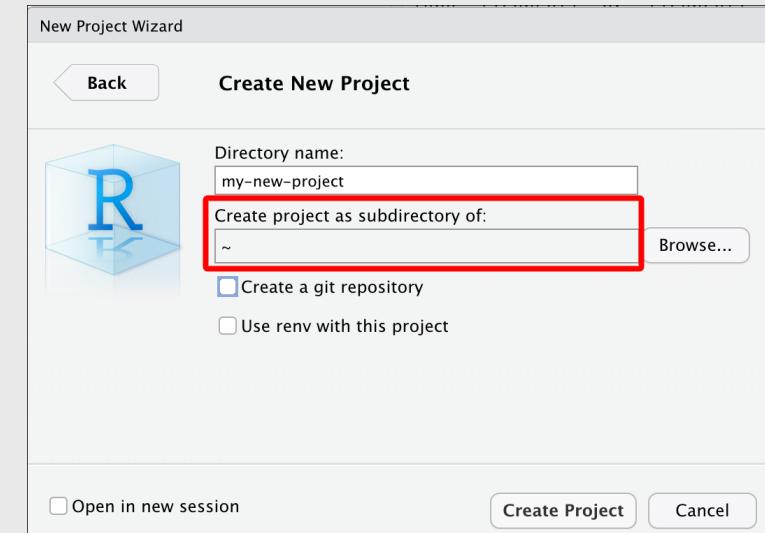


# Creating RStudio projects in new folder

## Create new folder name

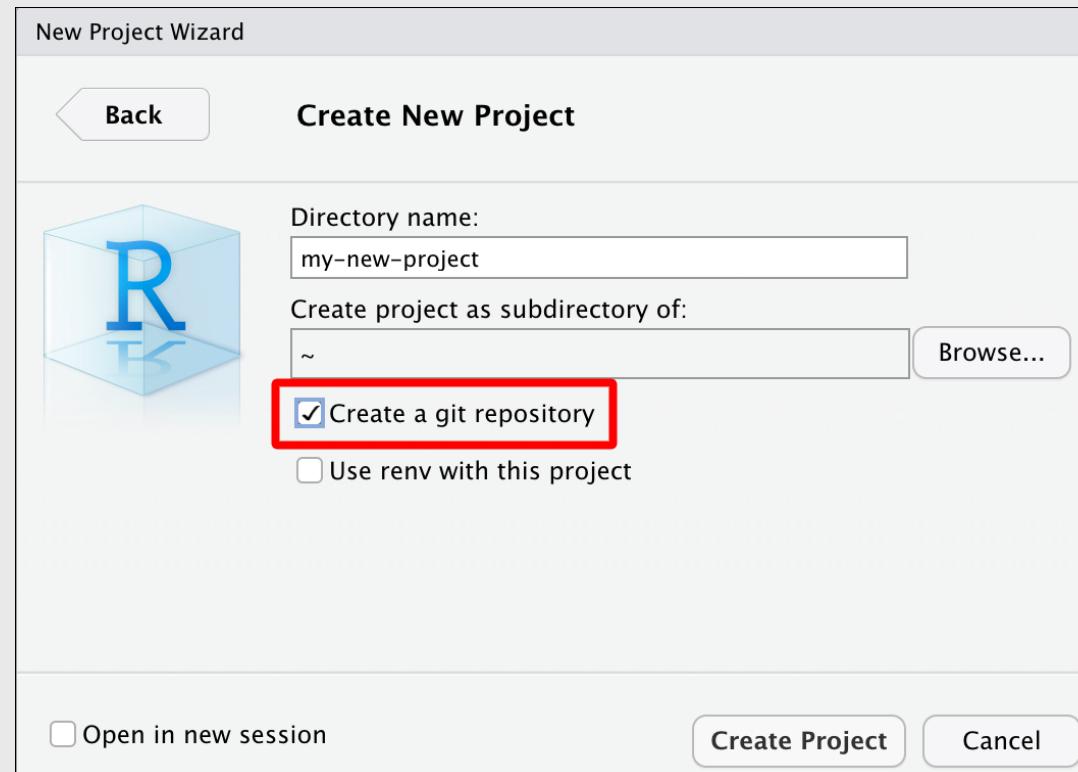


## Choose parent folder



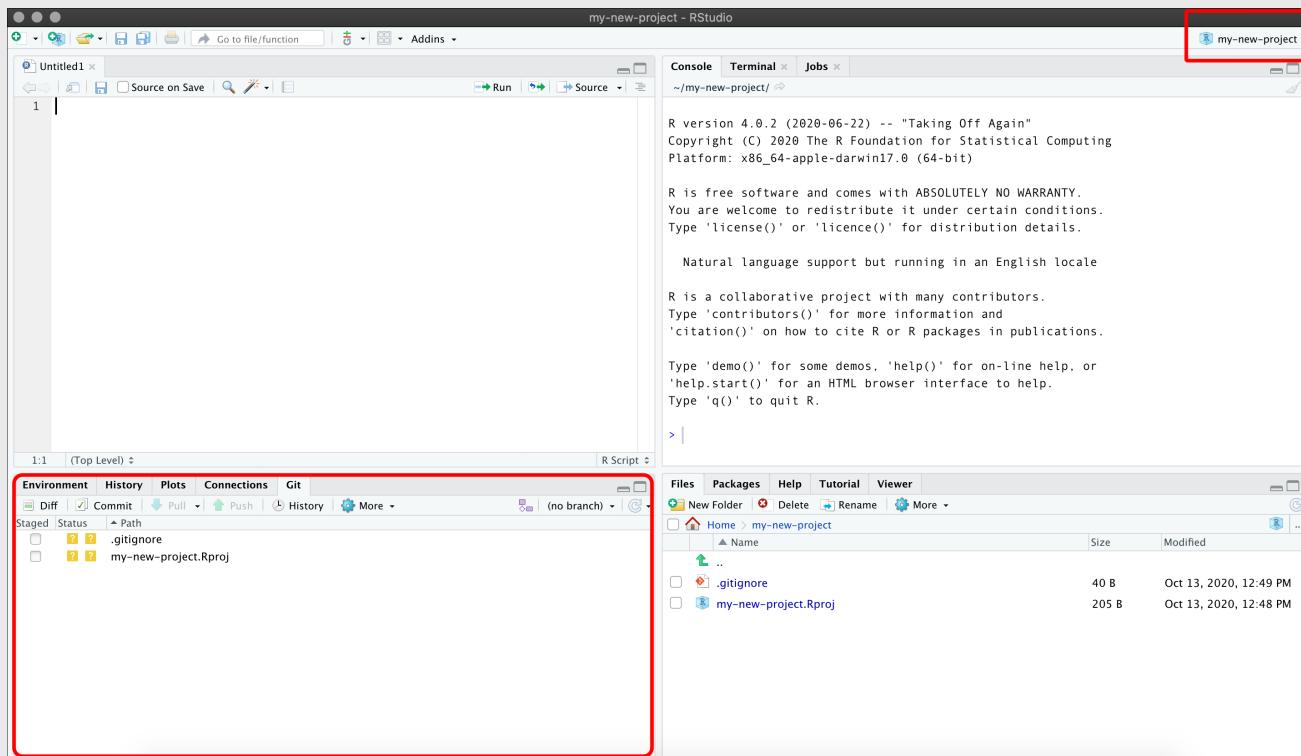
# Creating RStudio projects in new folder

If you have Git installed, select '*Create a git repository*'



# Creating RStudio projects in new folder

Check for new project name & Git pane



# Folder Structure

- separate raw and cleaned data
- keep documents and code separate
- keep figures separate
- name files appropriately (preferably 2 digit prefix)
- structure is reusable and easy to understand

```
project-name/
|--- CITATION
|--- project-name.Rproj
|--- README.md
|--- LICENSE
|--- requirements.txt
|---data/
|   |--raw/
|   |   |--raw-birds-data.csv
|   |--processed/
|   |   |--processed-birds-data.csv
|---doc/
|   |-- notebook.Rmd
|   |-- manuscript.Rmd
|   |-- changelog.txt
|---results/
|   |-- summarized-results.csv
|---code/
|   |-- 01-sightings-import.R
|   |-- 02-sightings-wrangle.R
|   |-- 03-sightings-model.R
|   |-- runall.R
```

# Folder structure

**Adapted from from 'Good enough practices in scientific computing'**

# Naming things

# Naming files

File names should be:

**1. human readable -> (makes sense)**

2020-10-12-270-301- **central-lab-metrics.csv**

**2. machine readable -> (regex)**

2020-10-12- **270-301**-central-lab-metrics.csv

**3. sort/order well -> (ISO 8601 date)**

**2020-10-12**-270-301-central-lab-metrics.csv

# Naming files\*

We can perform regular expression searches for files like this:

*Find 270-301 files*

```
grepl(pattern = "270-301",  
       x = "2020-10-12-270-301-central-lab-metrics.csv")
```

```
[1] TRUE
```

\*Adapted from [Jenny Bryan's slides](#)

# Naming files\*

Also acceptable: Logical order and underscores \_

```
files
```

```
[1] "01.0_import_lab-data.R"  "02.0_wrangle_lab-data.R"  "03.0_eda_lab-data.R"
```

```
stringr::str_split_fixed(string = files, pattern = "_", 3)
```

```
      [,1]   [,2]      [,3]  
[1,] "01.0" "import" "lab-data.R"  
[2,] "02.0" "wrangle" "lab-data.R"  
[3,] "03.0" "eda"     "lab-data.R"
```

\*Adapted from [Jenny Bryan's slides](#)

# File paths

# Use relative rather than absolute file paths

**Absolute paths** are specific to a system

`/project-name/data` → absolute path in macOS

`\\\project-name\\data` → absolute path in Windows

**Relative paths** are specific to a folder

`project-name/data` → relative path in macOS

`project-name\\data` → relative path in Windows

# Or use the `here` package

The `here::set_here()` function solves a lot of file path problems  
*(especially if you're not using R projects)*

```
library(here)
here::set_here(".")
list.files(all.files = TRUE, pattern = "here")
```

```
[1] ".here"
```

This creates a `.here` file (similar to `.Rproj` files)

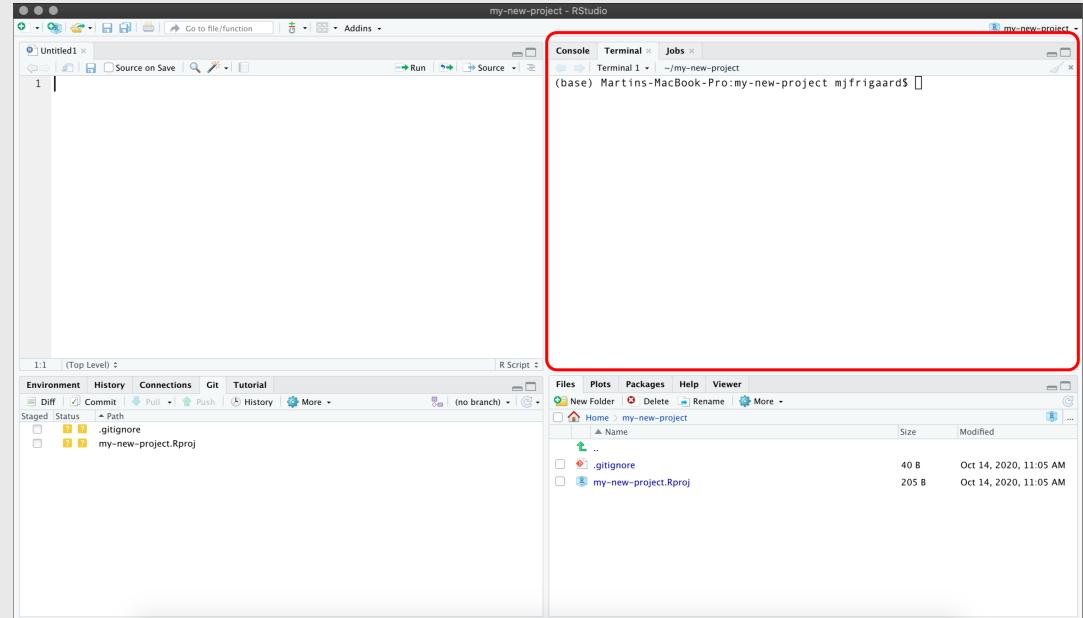
As long as the `.here` file stays in the referenced folder, you can include simply include `here::here()` in the top of your code files.

# Terminal pane

**Learn a handful of command-line tools  
to make life easier**

cd, pwd, mkdir, rm, ls, etc.

**RStudio comes with a Terminal pane for  
quick access to the command-line**



# Getting help

R comes with a *ton* of accessible help files

```
?read.csv
```

The screenshot shows the R Help browser interface. The title bar includes 'Files', 'Plots', 'Packages', 'Help' (which is selected), and 'Viewer'. Below the title bar, there are navigation icons and a search bar. The main content area shows the help page for 'read.table {utils}'. The page title is 'Data Input' and the subtitle is 'Description'. The description text reads: 'Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.' The 'Usage' section contains two code examples:

```
read.table(file, header = FALSE, sep = "", quote = "\"\"",  
          dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),  
          row.names, col.names, as.is = !stringsAsFactors,  
          na.strings = "NA", colClasses = NA, nrows = -1,  
          skip = 0, check.names = TRUE, fill = !blank.lines.skip,  
          strip.white = FALSE, blank.lines.skip = TRUE,  
          comment.char = "#",  
          allowEscapes = FALSE, flush = FALSE,  
          stringsAsFactors = default.stringsAsFactors(),  
          fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)  
  
read.csv(file, header = TRUE, sep = ",", quote = "\"\"",  
        dec = ".", fill = TRUE, comment.char = "", ...)
```

# Getting help online

R also has an incredible community! Click on the links below to see some of the common places for Q & A.

- 1) [Dedicated forum on RStudio Community](#)
- 2) [Questions tagged R on StackOverflow](#)
- 3) [Twitter topics with #rstats hashtag](#)

# Asking good questions (reproducible examples)

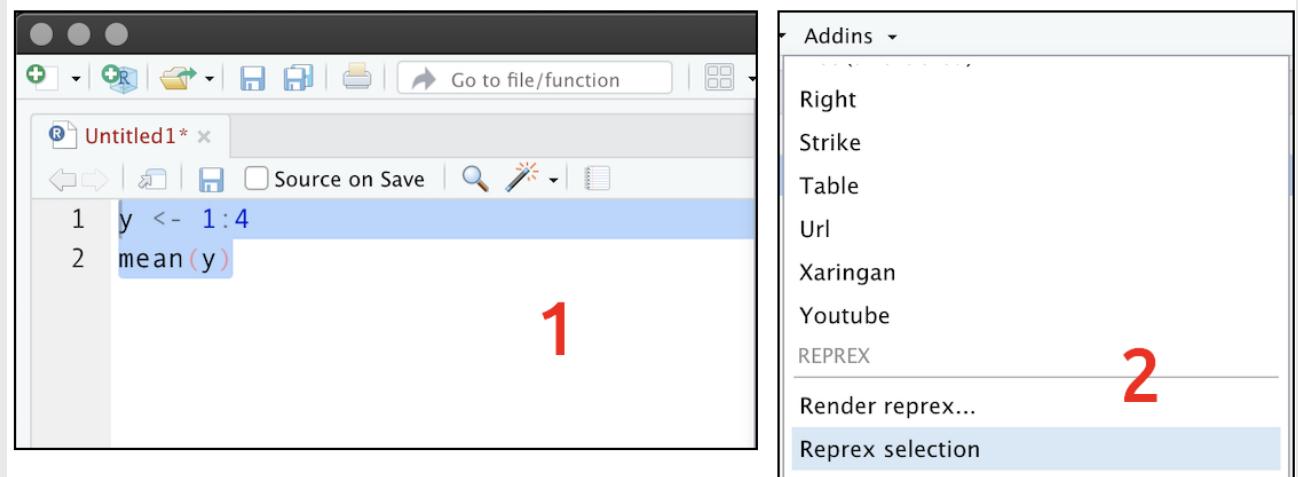
You'll get better results if you ask a question with a reproducible example. The [reprex package](#) was designed to help you create one!

```
install.packages("reprex")
library(reprex)
```

*Use the RStudio Addin to create a reproducible example from code you've copied onto your clipboard!*

# Reprex Addin 1

1. Copy code
2. Select Addin > Render selection



# Reprex Addin 2

1. Copy code
2. Select Addin > Render selection
3. Wait for console
4. Paste reprex

The screenshot shows the RStudio interface. The top pane is the Console, which displays the command `~/R/ ↵ Rendering reprex... Rendered reprex is on the clipboard.`. A red number '3' is positioned to the right of the console. The bottom pane is the Viewer, which shows the rendered reprex code:  

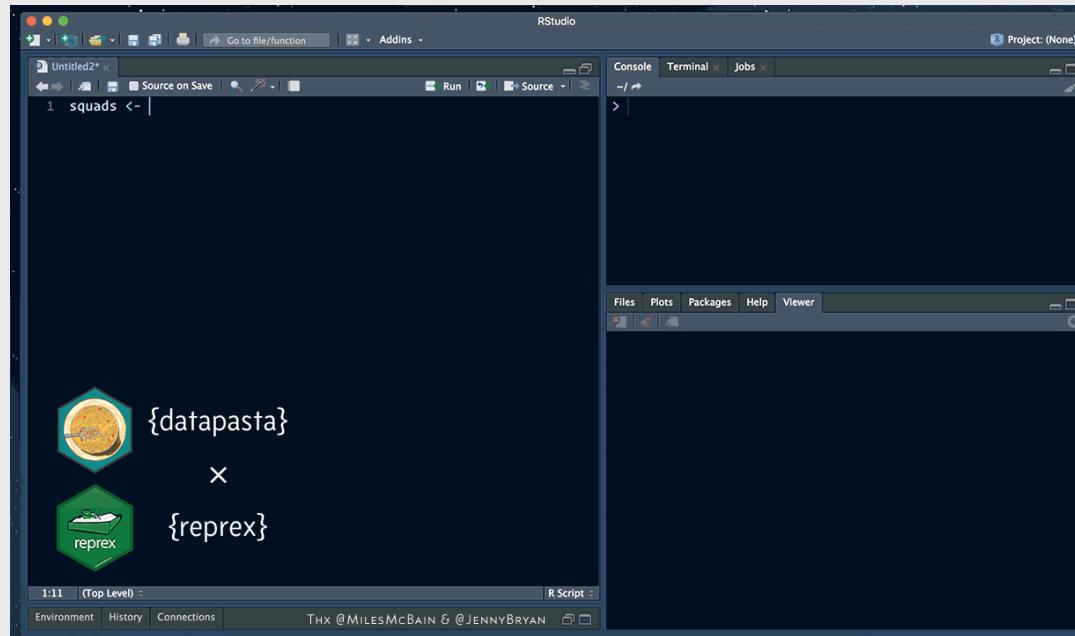
```
y <- 1:4  
mean(y)  
#> [1] 2.5
```

A red number '4' is positioned to the right of the viewer pane. At the bottom of the viewer pane, it says "Created on 2020-10-14 by the [reprex package](#) (v0.3.0)".

# Reprex + datapasta

To copy + paste actual data in a reproducible example, try **datapasta!**

<https://reprex.tidyverse.org/articles/articles/datapasta-reprex.html>



Learn more about R best practices:

1. [R for Data Science](#)
2. [Tidyverse](#)
3. [RViews Community Blog](#)

# THANK YOU!

## Feedback

@mjfrigaard on Twitter and Github

[mjfrigaard@gmail.com](mailto:mjfrigaard@gmail.com)